**AI-Powered Theme Customization for Yelo**

# 1. Objective

To implement an AI-based solution for automating and enhancing the customization of Yelo website themes. The goal is to allow users to create custom website themes by providing low-level instructions(prompts), according to the which the AI will generate custom HTML and CSS snippets.

# 2. Approach Options

A) Training A Model from Scratch

* **Description:**
  + Train a new model exclusively on Yelo's existing theme codes (HTML and CSS).
  + Model learns Yelo-specific coding patterns from the ground up.
* **Challenges:**
  + Requires a large, high-quality dataset.
  + Time-consuming and resource-intensive.
  + Risk of overfitting and limited generalization.
* **Requirements:** High cost and resource intensive

B) Fine-Tuning An Existing Pre-Trained Model

* **Description:**
  + Use a state-of-the-art model like OpenAI GPT-4, LLAMA 3.2, Deepseek, etc.
  + Fine-tune it on Yelo’s theme codes and customization examples.
* **Advantages:**
  + Leverages the model's prior knowledge of web development.
  + Requires less data and compute power.
  + Better generalization and adaptability.
* **Recommended** for efficiency and accuracy.

C) Agent-Based Setup with Fine-Tuned Model

* **Description:**
  + Build an agent-based system with modular components.
  + The agent handles user interaction, prompt generation, and code orchestration.
* **Advantages:**
  + Modular and flexible architecture.
  + Iterative refinement with feedback loops.
  + Scalable and easy to integrate with Yelo's platform.
* **Recommended** for modularity and long-term scalability.

# 3. Recommended Strategy: Fine-Tuning + Agent-Based Setup

Step 1: Data Collection & Preprocessing

* **Data Sources:**
  + Yelo’s existing HTML and CSS codebase.
  + Example theme customizations.
* **Preprocessing:**
  + Annotate data with component tags (e.g., header, footer).
  + Create prompt-response pairs.

Step 2: Fine-Tuning The Model

* **Model Selection:**
  + Use GPT-4, Llama 3.2, Deepseek, etc for code generation.
* **Fine-Tuning Steps:**
  + Prepare and clean the dataset.
  + Fine-tune the model on prompt-response pairs.
  + Validate the output quality using a test set.

Step 3: Agent Development

* **Agent Responsibilities:**
  + **Interpretation:** Parse user instructions.
  + **Decision-Making:** Identify target components (header, footer, etc.).
  + **Orchestration:** Query the fine-tuned model and generate code.

Step 4: Testing & Iteration

* **Unit Testing:**
  + Test with different customization instructions.
  + Validate output consistency.
* **User Testing:**
  + Deploy in staging environment.
  + Collect user feedback.
* **Iteration:**
  + Refine model and agent based on feedback.

Step 5: Deployment & Integration

* **Integration Steps:**
  + Deploy agent as a service.
  + Use Yelo’s API to integrate with the theme editor.
* **Monitoring:**
  + Track performance and log errors.
  + Use metrics to assess accuracy and user satisfaction.

**Step 4: Tech Stack & Resources Needed**

* **AI Model:**
* Pre-trained model: GPT-4 / Llama 3.2 / Deepseek / etc
* Fine-tuning framework: ANY
* **Infrastructure:**
* Database for storing customization data.
* GPU: NVIDIA GTX 3000/4000 series
* CPU: Standard multi-core CPU (Intel Core i7 or AMD Ryzen 7).
* Memory: 16-32 GB RAM (sufficient for model inference and light fine-tuning).
* Storage: SSD with moderate I/O throughput.
* **Development Tools:**
* FastAPI or Flask for the backend service.
* CUDA Development and Runtime Tools
* cuDNN

# 4. Blockers & Challenges

***Resource Constraints:***

* **Limited GPU power:** Yelo lacks access to high-end GPUs (like A100s or H100s), making large-scale fine-tuning and inference slower.
* **Limited RAM & storage:** May restrict the model's capacity to generate complex and optimized code snippets efficiently.

***Limited Training Data:***

* Yelo has a relatively small codebase for training compared to platforms like Shopify, resulting in fewer customization examples for fine-tuning.
* Smaller dataset may lead to lower model accuracy and generalization.

***API Limitations:***

* Yelo’s API may have limited flexibility for bulk or real-time theme updates, making dynamic customization slower or less efficient.
* Lack of granular control compared to Shopify's robust customization APIs.

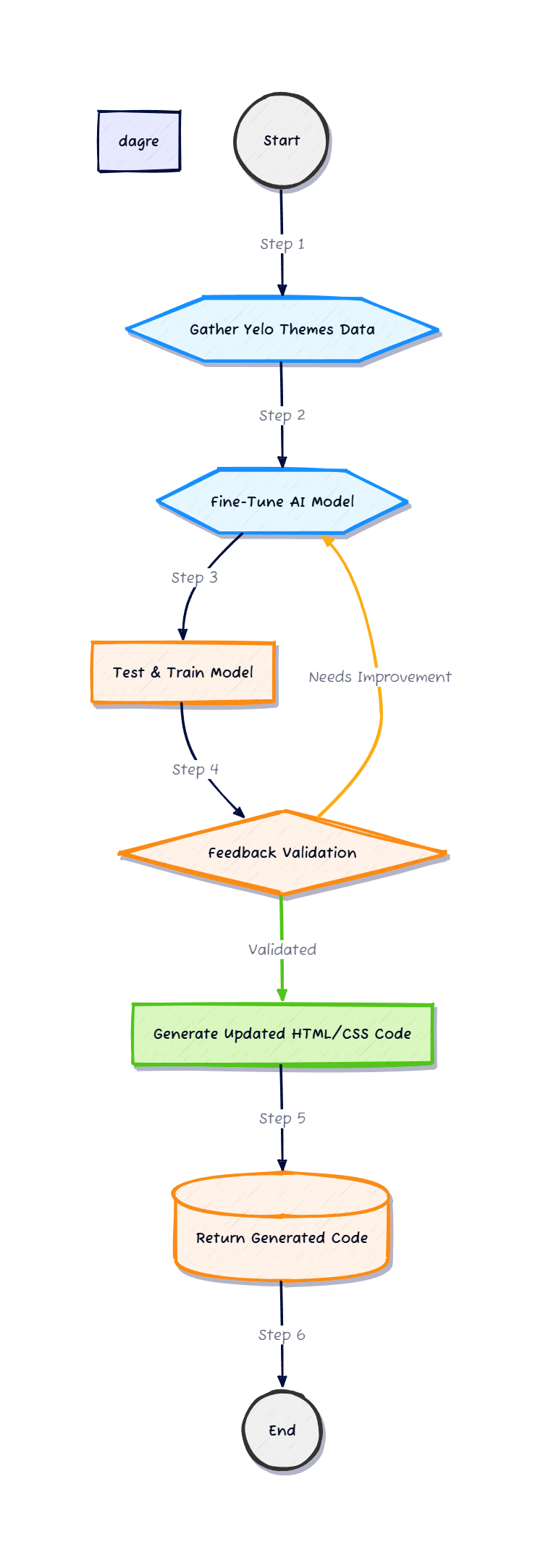
***Feature Gaps (Compared to Shopify):***

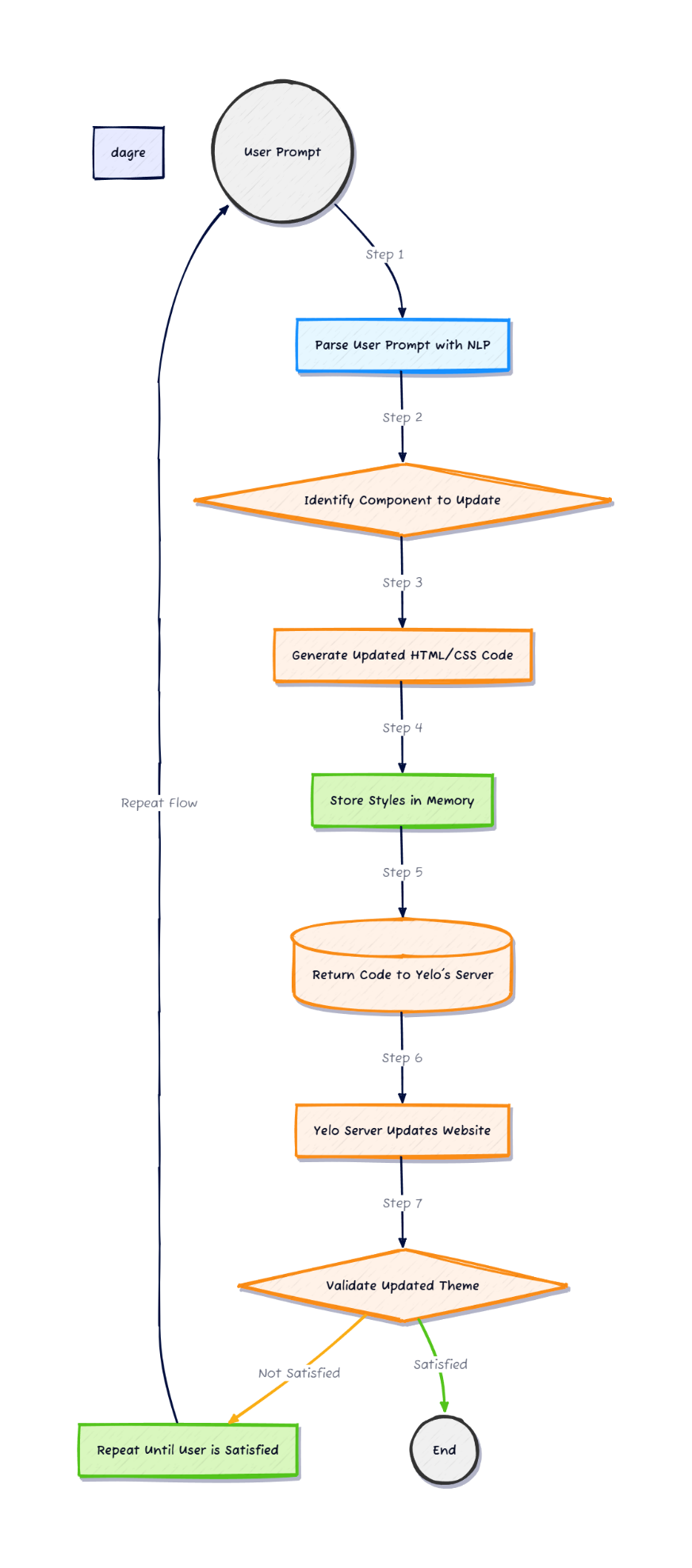
* **No Liquid Engine Integration:** Yelo lacks support for dynamic template languages like Liquid, limiting flexibility.
* **Fewer Pre-Built Components:** Shopify offers a larger set of reusable components, while Yelo relies on fewer, less dynamic blocks.
* **Limited Plugin Support:** Shopify has an extensive plugin ecosystem, whereas Yelo’s customization capabilities are relatively basic.

# 5. Conclusion

The recommended approach involves **fine-tuning a pre-trained model** on Yelo’s theme code and integrating it into an **agent-based setup**. This architecture provides flexibility, scalability, and efficiency, ensuring that the AI-powered theme customization system is robust and adaptable to future requirements. However, **resource constraints and limited data** may reduce the model's efficiency compared to larger platforms like Shopify, requiring iterative optimization and efficiency-focused improvements.

\*Check the next pages for preliminary architectural designs.

**Fine-tune Model**

**Theme Generation**